The Double Pouring Problem: A Lattice Approach

Nicolas Duhamel*

February 5, 2026

Abstract

The *double pouring problem* starts from k vessels containing integer amounts of water with total volume n. A single step selects two vessels with contents $x \leq y$ and replaces them by $2x$ and $y - x$; the objective is to make at least one vessel empty in as few steps as possible.

For $k = 3$, Frei et al. [1] gave an algorithm using $O((\log n)^2)$ steps and proved a lower bound of $\Omega(\log n)$.

We present a *semi-algorithm*: it may fail to terminate but whenever it terminates it does so in $O(\log n)$ steps.

It remains to prove that there exists an appropriate choice of initialization that turns the semi-algorithm into a genuine algorithm, i.e., guarantees termination on all instances.

1 Introduction

In the *double pouring problem* we are given k vessels containing integer volumes $a_1, \ldots, a_k \in \mathbb{N}$ with total volume $\sum_{i=1}^{k} a_i = n$. A *pouring step* chooses two vessels with contents $x \leq y$ and replaces them by

$$(x, y) \longmapsto (2x,\ y - x),$$

leaving the other vessels unchanged. The process preserves the total volume, and the goal is to reach a configuration in which at least one vessel is empty. We are interested in the worst-case minimum number of steps as a function of n.

We thank Ravi Boppana for introducing this problem to us; see also his presentation video [2].

The case $k = 3$ is probably the more challenging. Frei et al. [1] presented an algorithm requiring $O((\log n)^2)$ steps and proved a lower bound of $\Omega(\log n)$.

We give an algorithm that experimentally achieves $O(\log n)$ steps. The key idea is to move from the physical view (updating vessel contents) to a "dual" view that tracks integer relations among the volumes. We show that a pouring sequence can be encoded by a vector in a rank-2 lattice naturally associated with the state (a, b, c).

*nicolas.duhamel@laplateforme.io

2 Earlier work

We briefly summarize the results of Frei, Rossmanith, and Wehner [1] on what they call the *open pouring problem* (the case $k = 3$ of the double pouring problem). For a fixed total volume n, let $N(n)$ denote the worst-case minimum number of pouring steps over all initial configurations $(a, b, c) \in \mathbb{N}^3$ with $a + b + c = n$.

2.1 Bounds

Frei et al. [1] established logarithmic lower bounds and a quadratic-logarithmic upper bound.

Result	**Bound for** $N(n)$
Upper bound	$O((\log n)^2)$
Lower bound	$\Omega(\log n)$

Table 1: Asymptotic results for three vessels.

The main remaining question raised by this gap was whether the true order is $\Theta(\log n)$, $\Theta((\log n)^2)$ or in between.

2.2 Frei's rounds algorithm

Frei et al.'s strategy proceeds in *rounds*. Starting from an ordered state $0 < a \le b \le c$, each round transforms the configuration so that the new minimum vessel is at most $a/2$, using at most $\log n + O(1)$ pouring steps. After $O(\log n)$ rounds the minimum becomes 0, yielding a total of $O((\log n)^2)$ steps.

The key subroutine in a round is controlled by the ratio b/a. Let $p = \lfloor b/a \rfloor$ and $q = \lceil b/a \rceil$ and consider the remainders

$$r_1 = b - pa, \qquad r_2 = qa - b.$$

One always has $\min(r_1, r_2) \le a/2$. Using the binary expansion of p (or q), Frei et al. choose, at each step of the round, whether to pour from the second or third vessel into the smallest one. This forces the smallest vessel to double repeatedly while subtracting the appropriate amount from b (or c), and the round ends with the new minimum equal to r_1 or r_2.

A conceptual innovation of this approach (compared to earlier algorithm) is that each round divides the smallest vessel by 2, rather than reducing it by an additive constant.

3 The Lattice Formulation

This section explains how to encode pouring sequences as short vectors in an integer lattice. While the game is defined by local moves on vessel contents, the key invariant is linear: each configuration (a, b, c) determines a family of integer relations that remain compatible with the pouring operation.

3.1 A matrix viewpoint and invariants

Fix a state vector $\mathbf{v} = (a, b, c) \in \mathbb{N}^3$. A pouring step picks two coordinates, say $x \leq y$, and replaces them by $2x$ and $y - x$. Ignoring the ordering constraint $x \leq y$ for the moment, the update on a chosen pair is an integer-linear map. For instance, if we pour from the first vessel into the second (so (a, b) becomes $(2a, b - a)$), then

$$(a, b, c) \longmapsto (2a,\ b - a,\ c).$$

This can be viewed as the action of the following matrix.

$$M_{12} = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \in \mathrm{GL}_3(\mathbb{Z}).$$

Analogous matrices describe the other pair choices. Thus, abstractly, the reachable states from $\mathbf{v}$ lie in the orbit of $\mathbf{v}$ under a set of transformations, with the additional feasibility constraint that the chosen pair must satisfy $x \leq y$.

3.2 Algebraic reformulation

Let's consider coefficients $x, y, z \in \mathbb{Z}$, not all zero, such that

$$xa + yb + zc = 0. \tag{1}$$

The relevance of (1) is twofold: (i) it identifies a rank-2 lattice of relations intrinsic to the input instance, and (ii) the pouring operation corresponds to simple update rules on the coefficient vector when one insists that (1) remain satisfied.

3.3 The orthogonal lattice and its basic structure

Definition 1 (The orthogonal lattice)**.** Given $\mathbf{v} = (a, b, c) \in \mathbb{N}^3$, define

$$\mathcal{L}(\mathbf{v}) \ = \ \{(x, y, z) \in \mathbb{Z}^3 : xa + yb + zc = 0\} \ = \ \mathbf{v}^{\perp} \cap \mathbb{Z}^3.$$

The real orthogonal complement $\mathbf{v}^{\perp} \subset \mathbb{R}^3$ is a 2-dimensional subspace. Intersecting with $\mathbb{Z}^3$ yields a rank-2 sublattice $\mathcal{L}(\mathbf{v})$. Geometrically, $\mathcal{L}(\mathbf{v})$ is the set of all integer linear dependencies among the three vessel volumes.

A convenient explicit basis can be obtained from two "obvious" relations:

$$\mathbf{u}_1 = (b, -a, 0), \qquad \mathbf{u}_2 = (c, 0, -a),$$

since $\mathbf{u}_1 \cdot \mathbf{v} = ba - ab = 0$ and $\mathbf{u}_2 \cdot \mathbf{v} = ca - ac = 0$.

3.4 How pouring acts on coefficient vectors

The key link between pouring sequences and lattice vectors is that if a state update is given by a matrix $M \in \mathrm{GL}_3(\mathbb{Z})$, then integer relations transform by the inverse transpose. Indeed, if $\mathbf{v}' = M\mathbf{v}$ and $\mathbf{u}' = (M^{-1})^{\mathsf{T}}\mathbf{u}$, then

$$\mathbf{u}' \cdot \mathbf{v}' = ((M^{-1})^{\mathsf{T}}\mathbf{u}) \cdot (M\mathbf{v}) = \mathbf{u} \cdot \mathbf{v}.$$

Thus, maintaining the invariant $\mathbf{u} \cdot \mathbf{v} = 0$ throughout a sequence amounts to applying simple integer updates to $\mathbf{u}$. For the update M_{12} above one checks that

$$(M_{12}^{-1})^{\mathsf{T}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

so the induced map on the first two coefficients is

$$(x, y, z) \longmapsto \left(\tfrac{x+y}{2},\, y,\, z\right).$$

This operation is integral precisely when $x \equiv y \pmod 2$. Similar formulas hold for the other pairs. This explains the parity constraints that appear later in the algorithmic description.

In summary, a physically feasible pouring step in the physical space corresponds to selecting a pair of indices and applying an averaging map to the corresponding coefficients in the dual space, subject to an integrality (parity) condition.

3.5 The Lagrange–Gauss algorithm (rank 2)

Because $\mathcal{L}(\mathbf{v})$ has rank 2, we can work in a fixed basis and identify coefficient vectors with points of $\mathbb{Z}^2$. Equivalently, pick any integer basis $B = (\mathbf{b}_1, \mathbf{b}_2)$ of $\mathcal{L}(\mathbf{v})$ and write

$$\mathbf{u} = s\,\mathbf{b}_1 + t\,\mathbf{b}_2, \qquad (s, t) \in \mathbb{Z}^2.$$

Finding a good vector then becomes a two-dimensional shortest vector problem in the lattice generated by B. This is precisely the regime where lattice reduction is especially effective.

Let $L \subset \mathbb{R}^2$ be a two-dimensional lattice with basis vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^2$. The *Lagrange–Gauss* reduction algorithm repeatedly applies the following operations:

1. **Swap:** ensure $\|\mathbf{b}_1\| \le \|\mathbf{b}_2\|$ (otherwise exchange $\mathbf{b}_1$ and $\mathbf{b}_2$).
2. **Reduce:** set
$$\mathbf{b}_2 \leftarrow \mathbf{b}_2 - q\,\mathbf{b}_1, \qquad q = \left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2} \right\rceil,$$
where $\lfloor \cdot \rceil$ denotes the nearest integer.

This is the Euclidean algorithm applied to the Gram–Schmidt coefficient $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle / \|\mathbf{b}_1\|^2$.

Optimality in rank 2. A classical theorem due to Lagrange (1773) and Gauss (1801) states that the algorithm terminates with a *Gauss-reduced* basis $(\mathbf{b}_1, \mathbf{b}_2)$ satisfying

$$\|\mathbf{b}_1\| \le \|\mathbf{b}_2\|, \qquad |\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \le \tfrac{1}{2} \|\mathbf{b}_1\|^2.$$

In dimension 2, these conditions are equivalent to $\mathbf{b}_1$ being a shortest nonzero vector of the lattice, i.e., $\|\mathbf{b}_1\| = \lambda_1(L)$. Moreover, $\mathbf{b}_2$ is the shortest lattice vector that is linearly independent from $\mathbf{b}_1$.

Implication for the pouring problem. Since our orthogonal lattice $\mathcal{L}(\mathbf{v}) \subset \mathbb{Z}^3$ has rank 2, applying Lagrange–Gauss reduction to any basis of $\mathcal{L}(\mathbf{v})$ produces the shortest vector in the lattice.

4 The Lattice Algorithm

We now describe a concrete descent procedure in the dual space that matches the lattice viewpoint of Section 3. Unlike greedy strategies that operate directly on vessel contents, the algorithm maintains an integer relation $\mathbf{u} \cdot \mathbf{v} = 0$ and uses admissible pouring moves.

4.1 Guiding principle

Recall that a pour affecting vessels i and j corresponds (on the dual coefficients) to an averaging operation $u_i \leftarrow (u_i + u_j)/2$ (or the symmetric update), which is integral exactly when $u_i \equiv u_j \pmod 2$. The algorithm therefore enforces two constraints at every iteration:

1. **Physical admissibility:** the selected pair must satisfy $v_i \le v_j$ so that the updated content $v_j - v_i$ stays nonnegative.
2. **Parity:** the selected coefficients must satisfy $u_i \equiv u_j \pmod 2$ so that the induced dual update stays in $\mathbb{Z}^3$.

4.2 Algorithm

Informally, the algorithm iterates the following rule. At each step, we look for a *legal* move, i.e., a pair of indices (i, j) such that the corresponding dual coefficients have the same parity ($u_i \equiv u_j \pmod 2$), so that the induced averaging update remains integral.

Algorithm 1 Semi-Algorithm: Lattice descent (three vessels)

Require: Initial volumes $\mathbf{v} = (a, b, c) \in \mathbb{N}^3$
Ensure: A state with at least one empty vessel

Initialize: $\mathbf{u} \leftarrow (-b,\ a,\ 0)$
while all entries of $\mathbf{v}$ are nonzero **do**
  for each (i, j) with $i \neq j$ **do**
    if $v_i > v_j$ then **continue** ▷ **Physical check**
    if $u_i \not\equiv u_j \pmod 2$ then **continue** ▷ **Parity check**
    $t \leftarrow v_i$
    $v_i \leftarrow 2t;\ \ v_j \leftarrow v_j - t$
    $u_i \leftarrow (u_i + u_j)/2$
    break
  end for
end while
  return v

Proposition 1 (Halving of a coefficient gap)**.** Let $\mathbf{u} = (x, y, z) \in \mathbb{Z}^3$ with x, y, z pairwise distinct. At any step of the descent, if we perform a legal dual update averaging two coordinates (say $u_i \leftarrow (u_i + u_j)/2$ with $u_i \equiv u_j \pmod 2$), then the corresponding gap is divided by 2:

$$|u_i - u_j| \longmapsto \frac{|u_i - u_j|}{2}.$$

In particular, since x, y, z are all different, one of the three nonzero quantities $|x - y|$, $|x - z|$, or $|y - z|$ is divided by 2 at each step.

Proof. Assume we average the first two coordinates. Then $x' = (x + y)/2$ and $y' = y$, hence

$$|x' - y'| = \left|\frac{x + y}{2} - y\right| = \frac{|x - y|}{2}.$$

Because $x \neq y$, we have $|x - y| \neq 0$, so this is a genuine division by 2. The other choices of coordinate pairs are identical. □

Proposition 2 (A short step bound (semi-algorithm))**.** Consider the descent procedure as a semi-algorithm: it may enter an infinite loop (e.g., when it reaches a state where two of the coordinates are equal and nonzero), but if it terminates then it does so in less than $3\log_2(n)$ steps.

Remark 1. The algorithm is designed so that the invariant

$$\mathbf{u} \cdot \mathbf{v} = xa + yb + zc = 0$$

is preserved at every step. Consequently, reaching a dual vector with two zero coordinates is an immediate certificate that a vessel has been emptied. Indeed, if at some point we have $\mathbf{u} = (x, 0, 0)$ with $x \neq 0$, then the invariant gives $xa = 0$, hence $a = 0$. Similarly, $\mathbf{u} = (0, y, 0)$ implies $b = 0$, and $\mathbf{u} = (0, 0, z)$ implies $c = 0$.

Proof. Assume the semi-algorithm never reaches a bad state. Consider the dual vector $\mathbf{u} = (x, y, z)$ and define the quantity

$$\Delta(\mathbf{u}) = |x - y|\,|x - z|\,|y - z|.$$

As long as x, y, z are pairwise distinct, all three factors are nonzero, hence $\Delta(\mathbf{u}) \geq 1$. By Proposition 1, at each legal update one of the three gaps is divided by 2, and the other two gaps are unchanged; therefore $\Delta(\mathbf{u})$ is divided by 2 at every step where the coordinates remain pairwise distinct.

With the initialization $\mathbf{u}_0 = (-b, a, 0)$ we have

$$\Delta(\mathbf{u}_0) = |(-b) - a|\,|(-b) - 0|\,|a - 0| = (a + b)ab \leq n^3.$$

After t steps we thus have $1 \leq \Delta(\mathbf{u}_t) \leq n^3/2^t$, which implies $t \leq 3\log_2(n)$. □

4.3 Conclusion

The original goal of our approach was as follows: compute a shortest vector $\mathbf{u} \in \mathcal{L}(\mathbf{v})$ using Lagrange–Gauss reduction, and then use $\mathbf{u}$ to guide the pouring process via admissible dual updates.

In practice, however, this start from the shortest vector approach fails for some cases. In roughly 10% of randomly generated instances we observed a parity problem: we reach a configuration of the form

$$\mathbf{u} = (x, x, z),$$

where x and z have different parity. In that situation we reach an infinite loop, since any move involving the third coordinate violates the parity constraint, while averaging the first two coordinates leaves the state unchanged.

To mitigate this issue, we changed the initialization of the dual coefficients and used

$$\mathbf{u}_0 = (-b,\ a,\ 0)$$

instead. Experimentally, this choice eliminated the infinite-loop behavior on all instances we tested. At present we are unable to justify rigorously that this modified descent always terminates (i.e., that it never reaches a state of the form (x, x, z)).

While this is not a definitive proof of a $O(\log n)$ upper bound (due to possible infinite loops), it suggests that the approach is at least in the right direction. Our hope is that with a good initialization, the semi-algorithm never reaches the parity problem discussed earlier. Or maybe we can find a way to escape the infinite loop with another argument.

References

[1] F. Frei, P. Rossmanith, and D. Wehner. An open pouring problem. In *Proc. 10th International Conference on Fun with Algorithms (FUN 2021)*, pages 14:1–14:9, 2021.

[2] R. Boppana. *The Simple Problem That Stumped 99%* (video). YouTube, `https://www.youtube.com/watch?v=dI2RVGKqpCE`.